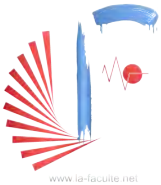
In the name of Allah, the Most Gracious, the Most Merciful



République Algérienne démocratique et populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

UNIVERSITE D'ALGER 1

Faculté de médecine d'Alger

Département de médecine

Objet : Révisions pour le rattrapage

Module : Informatique

Chef de groupe : Mohamed Abdelhamid Rouane, **Section :** D

Chers confrères et consœurs,

Après بسم الله الرحمان الرحيم Je tiens à vous présenter mes plus sincères salutations, je vous remercie aussi pour cette première expérience en tant qu'étudiants en médecine, nous avons encore 6 longues et fructueuse années devant nous, je dis bien 6 car le rattrapage n'est qu'une étape à passer et nous ne laisserons personne derrière nous, nous somme une promotion unie et solidaire et nous seront tous d'excellents médecins à l'avenir inchallah ! Bref je tenais à vous dire que cette étape qu'est le rattrapage sera pour vous une manière de vous " rafraîchir la mémoire " avant de passer en 2ème année, et afin de veiller au bon déroulement de cette épreuve moi et mes camarade sommes à votre disposition pour vous aider et vous conseiller, pour votre bien mais aussi pour le nôtre et afin de créer une communauté d'un haut niveau intellectuel qui redonnera ces lettres de noblesse à notre formidable religion !

Ceci étant dit, je ne puis abuser plus de votre temps, et donc je vous présente le cours que mon équipe et moi avons préparé en ce qui concerne le module d'informatique :

SOMMAIRE :

I. Introduction et définitions :

1) Introduction :

L'informatique est la science qui permet le traitement automatique de l'information d'où son nom **INFORMATIQUE** issu de la contraction de **INFORmation** et **autoMATIQUE**. Ce n'est pas la science des ordinateurs tout comme l'astronomie n'est pas la science des télescopes ! Cependant pour appliquer cette science il est nécessaire d'utiliser un ordinateur ou ces dérivés (tablettes tactiles…).

Les domaines d'application de cette science sont nombreux, je dirais presque infinis ! Mathématiques, jeux vidéo, biologie, physique, gestion, **médecine** et j'en passe…

Un système informatique se divise en 2 branches :

- Hardware : c'est l'aspect matériel du système qui représente les différents composants tel que les câbles, unités centrales, écrans… Etc.
- Software : c'est l'aspect logiciel, il régit l'ensemble des programmes informatiques tel que le système d'exploitation, les logiciels et les applications… Etc.

Composition d'un ordinateur :

- Eléments indispensables :
- Unité centrale : elle-même composée, entre autre, d'une carte mère, d'un processeur (CPU), de RAMs, d'un ou plusieurs disques durs, d'une ou plusieurs cartes graphiques / cartes son (optionnelles), d'une alimentation, et enfin pour contenir le tout un boitier aussi appelé châssis ou caisse.
- Ecran.
- Clavier et souris.
- Eléments complémentaires :
- Imprimante.
- Webcam.
- Disque dur externe.
- Périphériques de jeu tel que des manettes… Etc.

Certains éléments sont dits d'entrée, les périphériques d'entrée servent à commander l'appareil informatique ou à y envoyer des informations. Les données d'entrées sont numérisées pour pouvoir être utilisée par le processeur.

Exemples d'éléments d'entrée : Clavier, souris, USB (qui est considéré comme un élément d'entrée et de sortie !)... Etc.

Les autres éléments sont dits de sortie, les équipements de sortie servent à présenter les informations provenant d'un appareil informatique sous une forme reconnaissable par un humain.

Exemples d'éléments de sortie : Ecran, imprimante… Etc.

Afin d'optimiser l'utilisation de l'ordinateur, les informaticiens ont développés ce que l'on appelle **LE SYSTEME D'EXPLOITATION**, ce dernier constitue un support permettant de faciliter les interactions entre le matériel informatique et l'utilisateur (humain de préférence :p) au moyen d'une certaine interface.

2) Définitions : voici une liste des définitions utiles à connaitre en informatique :

- **Informatique :** (voir introduction).
- **Ordinateur :** Ou **computer** en anglais, est une machine permettant le travail avec les informations numériques appelées données.

A titre informatif : 1er ordinateur en 1830 par Charlie Bobbage

1er micro-ordinateur en 1971

- **Donnée :** Résultat directe d'une mesure.
- **Information :** c'est une donnée à laquelle un sens et une interprétation ont été donnés.
- **Connaissance :** Résultat de l'analyse et la réflexion du les informations, cette analyse est basée sur les expériences de la personne, ces idées…
- **Compétences :** Mise en pratique des différentes connaissances acquises.
- **Programme :** Séquence d'instructions écrites dans un langage précis que la machine peut comprend et qui vise à indiquer à l'ordinateur ce qu'il doit faire.
- **Processeur (ou Central Processing Unit : CPU) :** petite pièce en forme de carré, il représente le cerveau de l'ordinateur, il effectue des calculs à partir des données et exécute des taches appelées processus. Le processeur abrite d'autres détails tel que :

- Sa constitution : il est composé d'une unité de commande chargée de la lecture en mémoire et du décodage des instructions, et d'une unité de traitement qui est une unité arithmétique et logique chargée d'effectuer les calculs et d'exécuter les instructions.
- La fréquence qui désigne la vitesse d'un processeur, exprimée en GHz (exemple : un processeur cadencé à 4GHz peut effectuer 4Milliards d'opérations par seconde)
- Un processeur possède ce qu'on appelle des cœurs leurs nombre détermine le nombre de taches pouvant être effectués selon la vitesse des cœurs (exemple : un processeur quad-core cadencé à 4GHz désigne un processeur pouvant effectuer 4Milliards d'opérations par cœur, c'est-à-dire dans notre exemple 16Milliards d'opérations)

Comme vous l'avez remarqué, pour un processeur a 4 cœurs il est appelé **quad-core**, un processeur à 6 cœurs est appelé **hexa-core**, un à 8 cœurs **octa-core**, un à 2 cœurs **dual-core,** on les appelle des processeurs **multi-cœurs**, quand à un processeur standard a un seule cœur, il est dit **single-core.**

- Mémoire cache : une petite mémoire très rapide permettant au processeur d'effectuer des taches peut volumineuses rapidement.
- **Mémoire (Centrale) :** ou mémoire vive appelée aussi **RAM** pour **Random Access Memory**, elle permet de stocker momentanément les données et les instructions et les transmettre au processeur pour qu'il puisse les calculer et gérer.
- **Horloge :** Rythme le travail du processeur et de la mémoire a une certaine fréquence (généralement en MHz pour la mémoire et en GHz pour le CPU).
- **Mémoire disque :** espace de stockage lente en comparaison avec la mémoire cache du processeur et la mémoire vive, mais elle est permanente (ROM).
- **Périphériques :** ils représentent les organes du pc permettant la communication avec le couple CPU-RAM (exemples : clavier, souris, écran, imprimante, USB…).
- **Logiciel :** partie software, no tangible, s'exécute dans la RAM, c'est un ensemble de séquences d'instructions interprétables par la machine, il détermine donc d'une certaine manière les taches à effectuer par l'ordinateur.

Il existe plusieurs types de logiciels qu'on peut classer principalement en 2 groupes :

a. Les systèmes d'exploitation (Operating System : OS) : déjà abordés précédemment, le système d'exploitation est stocké dans le disque dur et exécutable, comme tous les logiciels, dans la mémoire vive.

- Exemples d'OS : Windows (de 95 à 10 qui est la dernière version sortie récemment), MacOS développé par Macintosh de la firme Apple, Linux qui est un système open source et gratuit.

b. Les logiciels d'applications : Ce sont des programmes exécutables permettant de réaliser différents types de fonctions (retouches d'images, édition de vidéos, navigation internet, jeux…) par exemple : Photoshop. Les applications sont gérées par le système d'exploitation dont elles sont spécifiques (compatibles).

R ! Il manque certaines définitions que nous aborderons plus tard, nous voulons que chaque chose soit dans son contexte ☺

II. Système binaire, pixels, et tout ce qu'il faut savoir sur les chiffres et les calculs à utiliser dans ce module :

➢ Le système binaire :

A la base, les ordinateurs ne savent compter que 0 ou 1 correspondant respectivement aux états de passage du courant dans le système de sorte que 0 c'est OFF et 1 c'est ON.

En mathématique, tout comme dans notre vie de tous les jours, nous utilisons la **base décimale** pour nos calculs, c'est-à-dire qu'on possède 10 symboles représentant la base de nos numéros et qui sont 0,1,2,3,4,5,6,7,8,9 une fois arrivés à 9 nous jouons sur l'ordre et la disposition des numéros afin de créer un nouveau numéro de sorte que le numéro qui se trouve à gauche est multiplié par 10^(sa position), c'est un peu flou de prime abord je vous l'accorde, donc je vais tenter d'éclaircir un peu la chose par quelques exemples : 10 = 0*10^0 + 1*10^1

100 = 0*10^0 + 0*10^1 + 1*10^2

10110 = 0*10^0 + 1*10^1 + 1*10^2 + 0*10^3 +1*10^4

Je crois que vous avez compris le principe ! (si oui, alors faites-moi donc cet exemple 1298 :3).

R !

- On nous donne toujours des exercices beaucoup plus difficiles que les exemples du cours (POURQUOI !!!).
- Les chiffre de 2 à 9 sont eux aussi *10^(leur position).

Voici un autre exemple pour illustrer ça : 789 = 9*10^0 + 8*10^1 + 7*10^2 (voilà comme ça vous pourrez faire l'exemple d'en haut sans problème normalement ;)

Maintenant que vous avez compris la base décimale, il faut savoir qu'il existe plusieurs bases :

La base de 60 (sexagésimale) est celle utilisée pour les heures et les minutes de sorte qu'une fois arrivés à 60 on a +1,

La base 24 est utilisée pour les heures dans un jour,

Ce qui nous intéresse nous en informatique c'est **le binaire** (*ah enfin les choses sérieuses, je peux commencer à lire le cours maintenant*)

Pour le système binaire c'est le même principe que pour le décimale (et oui vous avez déjà compris le binaire sans le savoir) sauf que cette fois ci la base de calcul c'est 2, donc on n'a que 2 symboles à utiliser et qui sont le 0 et le 1 les autres numéros sont basés sur l'ordre et l'agencement des 0 et des 1, mais au lieu de multiplier par 10 on multiplie par 2^(la position).

R !

- La 1ere position est 0 pas 1 (vous le remarquerez dans les exemples).
- Il faut impérativement respecter l'ordre avec lequel se fait le calcul, un changement d'ordre implique un changement de numéro ! Vous allez découvrir cet ordre par vous-mêmes à travers les exemples.
- Les numéros impaires se finissent toujours par 1 et les numéros paires par 0 (101=5 il est impaire, 110=6 il est paire, la conversion est faite du binaire au décimal).

Exemple : 101101 (en binaire) = 1*2^0 + 0*2^1 + 1*2^2 + 1*2^3 + 0*2^4 + 1*2^5 = 45 (en décimal)

Donc pour aller du binaire au décimal il c'est une suite de multiplications *2^(la position) puis on additionne le tout comme dans l'exemple précèdent, c'est aussi ce qu'on va faire dans les exemples suivants :

11010 = 0*2^0 + 1*2^1 + 0*2^2 + 1*2^3 + 1*2^4 = 26

111 = 1*2^0 + 1*2^1 + 1*2^2 = 7

10 = 0*2^0 + 1*2^1 = 2

1001 = 1*2^0 + 0*2^1 + 0*2^2 + 1*2^3 = 9

Exercice : Convertir les numéros suivants du binaire au décimal : 01, 101, 1100 ; 1010101001010110…

A présent nous savons allez du binaire au décimal, il faut maintenant apprendre à faire le chemin inverse, et l'inverse de la multiplication c'est la division, donc pour passer du décimal au binaire il faut passer par une suite de divisions /2, on divise continuellement les résultants des divisions précédentes jusqu'à obtenir 0, puis il faut écrire les restes des divisions selon un certain ordre bien précis pour obtenir notre numéro en binaire, vous allez mieux comprendre ce charabia à travers les exemples suivants :

10 =

10/2 = 5 reste 0

5/2 = 2 reste 1

2/2 = 1 reste 0

1/2 = 0 reste 1

L'ordre avec lequel on écrit notre numéro en binaire est le suivant : on va lire notre numéro de bas en haut puis l'écrire de gauche à droite comme ceci : 1010

Donc au final on a : 10 (en décimal) = 1010 (en binaire), enchainons avec d'autres exemples :

5 = 101 (je l'ai fait directement car on a déjà fait les divisions de 5 pour l'exemple de 10)

17 =

17/2 = 8 reste 1

8/2 = 4 reste 0

4/2 = 2 reste 0

2/2 = 1 reste 0

1/2 = 0 reste 1

= 10001

26 =

26/2 = 13 reste 0

13/2 = 6 reste 1

6/2 = 3 reste 0

3/2 = 1 reste 1

1/2 = 0 reste 1

= 11010

12 =

12/2 = 6 reste 0

6/2 = 3 reste 0

3/2 = 1 reste 1

1/2 = 0 reste 1

= 1100

Ça devrais suffire comme exemples, vous pourrez toujours vous amuser à convertir n'importe quel numéro qui vous passe par la tête en binaire.

On a appris ensemble c'est quoi le binaire, comment le convertir en décimal et vice versa, mais ça sert à quoi concrètement ?

L'ordinateur, comme nous l'avons déjà découvert lors de l'introduction, est une machine capable de traiter et de stocker numériquement les informations, mais la machine ne comprends qu'en binaire voyez-vous, donc afin de traiter ces informations, la machine fait continuellement des conversions et stocke les informations sous forme binaire dans un espace mémoire appelé **bit**.

- **Les bits :**

Le terme **bit** (b en minuscules !) signifie **binary digit**, c'est la plus petite unité d'information manipulable par la machine.

Les bits signifient les états possibles en traduisant les instructions sous forme de 0 et de 1 soit en binaire, de ce fait on a :

1 bit signifie 2 états 0 ou 1

Avec 2 bits on obtient 4 états différents : 00, 01, 10, 11

Avec 3 bits on obtiendra 8 états : 000, 001, 010, 100, 011, 110, 101, 111

On remarque que le nombre d'états possibles c'est 2^(nombre de bits)

1bit = 2^1 = 2 états

2bits = 2^2 = 4 états

3bits = 2^3 = 8 états

Dans un nombre binaire la valeur d'un bit est appelée **poids**, ce dernier dépend de la position du bit de droite à gauche, c'est le même principe que pour les conversions du binaire qu'on a fait tout à l'heure, donc pour convertir un mot binaire en décimal, il faut multiplier chaque bit par son poids puis additionner chaque résultat

Exemple :

Nombre binaire	1	0	1	1	0	1	0	0	1
Poids	1*2^8 = 256	0*2^7 = 0	1*2^6 = 64	1*2^5 = 32	0*2^4 = 0	1*2^3 = 8	0*2^2 = 0	0*2^1 = 0	1*2^0 = 1

Pour la conversion elle se fait de la même façon que pour nos exemples sus-cités, mais on en faire quelques-uns ici :

1011 = 1*2^0 (ce 2^0 est le poids de ce bit) + 1*2^1 + 0*2^2 + 1*2^3 = 11

011010 = 0*2^0 + 1*2^1 + 0*2^2 + 1*2^3 + 1*2^4 + 0*2^5 = 28

En informatique, un total de **8bits** et dit **Octet** (ou **Byte** en anglais avec un B en majuscule !)

Donc 1Octet = 8bits = 2^8 états

L'Octet est une unité de stockage de l'information (elle peut désigner le volume d'un fichier ou la capacité de stockage d'un disque).

Pour 1 Octet, le plus petit nombre est 0 (représenté par 00000000), le plus grand est 255 (représenté par 11111111) donc ce qui fait 2^8=256 possibilités de valeurs différentes.

Pour 2 Octets c'est 2^16 et pour 3 Octets c'est 2^24 (c'est 2^le nombre de bits contenus dans ces octets)

Tout comme pour les autres unités, on peut avoir à faire des conversions, ces dernières et à la différence des autres unités qui sont régies par le système décimal, en informatique les unités sont conditionnées par le système binaire de sorte que :

Unité	**Appellation**	**Conversion en *octets***	**Conversion en *bits***
1 o	1 octet	1	8
1 Ko	1 Kilo octet	1024 = 2^10	8192
1 Mo = 1024 Ko	1 Méga octet	1048576 = 2^20	8388608
1 Go = 1024 Mo	1 Giga octet	1073741824 = 2^30	8589934592
1 To = 1024 Go	1 Téra octet	1099511627776 = 2^40	8796093022208

- **Le code ASCII :**

ASCII = American Standard Code for Information Interchange, autrement dit c'est un code Américain standard pour l'échange d'informations, il représente un équivalent numérique des différents caractères (lettre, chiffres,…)

A la base il représente les caractères de 7bits (128 caractères de 0 à 127)

Le code ASCII se présente sous cette forme :

Esp = 32	0 = 48	A = 65	[= 91	a = 97	{ = 123
! = 33	1 = 49	B = 66	\ = 92	b = 98	| = 124
'' = 34	2 = 50	C = 67	] = 93	c = 99	} = 125
# = 35	3 = 51	D = 68	^ = 94	d = 100	.
$ = 36	4 = 52	E = 69	_ = 95	e = 101	.
% = 37	5 = 53	F = 70	.	f = 102	.
& = 38	6 = 54	G = 71	.	g = 103	
' = 39	7 = 55	H = 72	.	h = 104	
(= 40	8 = 56	I = 73		i = 105	
) = 41	9 = 57	J = 74		j = 106	
		.		.	
		.		.	
		.		.	
		Z = 90		z = 122	

Tous ce que l'on peut vous demander sur cette partie du cours c'est soit des définitions soit vous demander de traduire un mot en ASCII à l'aide d'un tableau donné. Exemples : Vie = 87 105 101

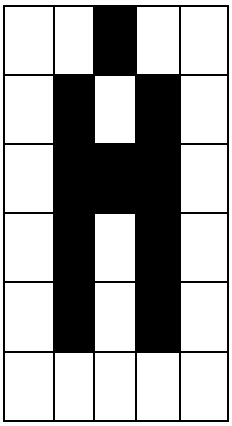
Chef = 99 104 101 102

R !

- Le code ASCII étant limité a 128 caractères il fut nécessaire de créer un code avec plus de caractères, ce qui donna naissance à l'**UNICODE** qui est stocké sur 2 octets donc 2^16=65536 caractères possibles !

- **Affichage des images :**

Une image numérique est constituée d'un ensemble de petits carrés appelés **Pixels**, ce dernier est une contraction de **Picture Elements** qui veut dire que le pixel est la plus petite unité constitutive d'une image, l'ensemble des pixels d'une image est contenu dans une sorte de tableau a 2 dimensions. Exemple :



5 pixel * 6 pixel

La définition d'une image est le nombre de pixels constituants l'image, elle est écrite comme étant le produits du nombre de pixels en largeur et le nombre de pixels en hauteur. Exemples :

- Une image de définition 800*600 est une image qui a 800 pixels en largeur et 600 en hauteur

- Une image de 640 pixels en largeur et 480 en hauteur a une définition de 640*480

La résolution quand a elle, c'est le nombre de pixels c'est le nombre de pixels par unité de surface, sachant que la surface d'un écran se mesure en pouces, la résolution d'une image est considérée en **Pixels Par Pouce (PPP ou DPI pour Digit Per Inch)**. Exemple : une image de 300PPP signifie qu'elle est constituée de 300 pixels par rangée et par colonne, cela signifie aussi 90000 pixels sur un pouce carré.

A titre informatif : La résolution de référence de 72 DPI qui nous donne 1''/72 (1 pouce divisé par 72) soit 0.353 mm (pour être tout à fait sincère je pense que c'est une information superflue il ne faut pas s'attarder dessus, je l'ai juste mentionné au cas où ☺).

On sait maintenant que l'image est représentée par un tableau a 2 dimensions dont chaque case est un pixel, la représentation en binaire d'une image dépend du poids de celle-ci.

Le poids d'une image est calculé, en **Octets**, en fonction du nombre de pixels qui constituent l'image (c'est-à-dire sa définition) et de la couleur des pixels (intensité des pixels), or le poids de l'image c'est le produit des 2 paramètres précédemment cités

Pour calculer **le poids d'une image** il faut donc connaitre **sa définition** et **le codage couleur** qui lui est associé, ce codage couleur se résume ainsi :

- **Noir et blanc :** Pour ***1bit*** dans chaque case il est possible d'obtenir ***2 couleurs***.
- **Image en 16 couleurs ou 16 niveaux de gris :** Pour ***4bits*** dans chaque case on obtient ***16 couleurs*** qui représentent ***16 dégradés de gris*** allant du noir au blanc.
- **Image en 256 couleurs :** Stocké sur ***1octet (8bits par case)*** il permet d'obtenir ***256 couleurs*** (un spectre de couleurs ou spectre de gris).
- **Image en 16 millions de couleurs (true color ou RGB pour Red Green Blue) :** c'est une décomposition des couleurs à l'aide des 3 couleurs primaires (rouge, vert et bleu), il est stocké sur ***3octets (donc 24bits par case)***.

Donc le poids d'une image (en bits !) c'est :

- Sa définition * 1 si l'image est en noir et blanc
- Sa définition * 4 si l'image est en 16 couleurs
- Sa définition * 8 si l'image est en 256 couleurs
- Sa définition * 24 si l'image est en true color.

R ! Pour calculer le poids d'une image ne Octets il faut multiplier par le codage couleur en octets !

Exemple : pour calculer le poids d'une image d'une définition de 640*480 en true color on va procéder comme ce qui suit :

640*480 = 307200

True color = 24 bits = 3 octets

Le poids de l'image en bits = 307200*24

Le poids de l'image en octets = 307200*3 = 921600 octets = 900*1024 = 900 Ko

R ! On calcule généralement le poids d'une image en **Octets**.

III. Algorithmique :

a) Définition :

Un algorithme est une suite organisée d'opérations (instructions) élémentaires qui permet de résoudre un problème donné (par exemple additionner 2 nombres réels).

Un algorithme une fois écrit dans un langage de programmation précis va nous donner un programme informatique.

b) Structure d'un algorithme :

Un algorithme s'écrit selon une structure bien précise et varie selon le problème à résoudre, cette structure est présentée comme ceci :

```
Algo [titre] ;

Const [nom de la constante] = valeur de la constante

Var [déclaration des variables : type de variables] ;

Début




Fin
```

R ! Le vide qui se trouve entre le Début et la Fin représente le corps de l'algorithme, il peut être : des affectations, des tests, des boucles, des opérations numériques… Etc.

Ne vous inquiétez pas nous allons tout expliquer pas à pas et essayer de construire et de lire des algorithmes à travers des exemples ;)

- **Le titre :** Il se compose du mot Algo suivi du titre que vous voulez donner à votre algorithme, le titre doit être aussi abrégé et significatif que possible (vous allez mieux comprendre à travers les exemples plus tard, mais voici quand même un exemple concernant le titre : Algo Add désigne dans cet exemple le titre d'un algorithme qui effectue une addition)
- **Les Constantes et les variables :**

i. ***Les Constantes :*** Elles commencent par le mot Const suivi par le nom de la constante puis ça valeur (exemple : ***Const Pi = 3.14***). Les constantes sont rarement utilisées, je vous suggère de ne pas trop vous baser sur ça.

ii. ***Les variables :*** Elles commencent par le mot Var suivi du nom de la ou les variables puis le type de la ou les variables (exemple : Var a, b : Entiers c : Réel).

R ! Les constantes sont déclarées avant les variables.

Les variables sont de 3 types :

1. ***Variables numériques :*** Ce sont les différents numéros avec lesquels on peut effectuer une opération (un calcul), les numéros peuvent être des entiers ou des réels.
2. ***Variables alphanumériques :*** Elle peuvent être des lettres, des nombres, des symboles… Etc. Elles sont soit :

- Des caractères : lettre, chiffre, numéro ou symbole isolés (exemple : Var a, b, c, 1, 2, 3, ! : caractères).
- Des chaines de caractères : Elles représentent une succession de plusieurs caractères, comme pour un mot ou un numéro a plusieurs chiffres (exemple : Var mot : str)

R ! La notation str est utilisée pour désigner une variable en chaine de caractères (str signifie string en anglais qui veut dire chaine de caractères).

Les variables alphanumériques s'écrivent entre guillemets dans l'algorithme comme ceci : '' chaine ''

3. ***Variables de Bool (booléens) :*** Ce sont des variables qui ne désignent que les valeurs **vrai** ou **faux** (**true** or **false**) soit 1 ou 0. Exemple : Var True, False : Bool ;

➢ **Les instructions de base :**

i. ***La lecture :*** Cela signifie que l'utilisateur va entrer une donnée qui va être lu par l'ordinateur, par exemple lire une certaine variable en l'entrant à l'aide du clavier. Exemple : Lire (a, b) ce qui signifie qu'on a entré les valeurs a et b (notamment à l'aide du clavier ou de la souris).

ii. ***L'écriture :*** Ou instruction d'affichage, c'est une instruction qui permet à notre ordinateur d'afficher à l'écran ce que l'on veut. Exemple : Ecrire ('' Entrez une valeur '')

Dans notre exemple l'ordinateur va afficher : Entrez une valeur.

iii. ***L'affectation :*** C'est une instruction qui permet d'affecter une valeur a une variable pour faire des calculs par exemple (nous allons voir ça bientôt dans les exemples d'algorithmes)

L'affectation suit l'ordre logique qu'on lui impose. Exemple :

A=1 et B=2 et C=3

A ← B

B ← C

C ← A

On aura au final A=2, B=3, C=2, nous avons affecté B à A et C à B et A (à qui on a déjà affecté B) à C

Cependant avec notre affectation précédente nous avons perdu la valeur de A, et donc pour conserver cette valeur nous allons devoir passer par une variable temporelle qui nous permettra de permuter entre nos valeurs sans en perdre une en chemin :

D ← A

A ← B

B ← C

C ← D

On obtient à la fin A=2, B=3, C= 1, D=1 (toutes nos valeurs sont la).

iv. ***Les tests :*** Ou ***instructions conditionnelles***, ce sont des conditions à vérifier, ils se font selon la structure suivante :

Si (condition) Alors (conséquence)

Sinon (conséquence de la condition contraire)

Mais il se pourrait que nos tests soient longs ou qu'on ait à faire des tests imbriqués (un test dans un test) il serait préférable d'écrire chaque test avec son début et ça fin comme ceci :

Si

Début si

(Condition) Alors (conséquence)

Fin si

Sinon

Début sinon

(Conséquence de la condition contraire)

Fin sinon

R ! La structure ''sinon'' est une structure alternative, on peut s'en passer et on ne l'utilise que pour ne pas trop charger notre algorithme de conditions et pour éviter les répétitions.

v. ***Les boucles :*** Ce sont des instructions itératives (répétitives) qui permettent de répéter une opération un certain nombre de fois (pour éviter de la répéter nous-même a chaque fois)

Les boucles sont de 3 sortes :

1. Tant que (condition) Alors (exécuter la tache)
2. Pour (condition) Alors (exécuter la tache)
3. Répéter (la tache) jusqu'à (condition d'arrêt)

Maintenant il est temps de traiter quelques exemples d'algorithmes afin d'avoir bien en tête les principes que nous avons appris, et afin d'apprendre à lire et à construire un algorithme. (Je laisserais des remarques à la fin pour éclaircir certains points qui me semblent être sujets à d'éventuelles questions)

- **Exemple 1 :** Ecrire un algorithme qui permet de calculer la surface d'un rectangle.

```
Algo surface ;
Var S, L, l : réels ;
Début
 Ecrire (''Entrez la longueur L puis la largeur l'') ;
Lire (L, l) ;
S ← L*l
Ecrire (''La surface du rectangle est :'', S) ;
Fin
```

- **Exemple 2 :** Ecrire un algorithme qui détermine la valeur maximale entre 3 variables entières.

```
Algo Max3 ;
Var a, b, c, Max3 : entiers ;
Début
Ecrire (''Donnez les valeurs des variables entières a, b et c'') ;
Lire (a, b, c) ;
Si a>b Alors
        Si a>c Alors Max3 ← a
        Sinon Max3 ← c
Sinon
        Si c>b Alors Max3 ← c
        Sinon Max3 ← b
Ecrire (''La valeur maximale est :'', Max3) ;
Fin
```

- **Exemple 3 :** Ecrire un algorithme qui permet de calculer le PGDC entre deux entiers A et B en utilisant la méthode d'Euclide.

```
Algo PGDC ;
Var A, B, R : entiers ;
Début
Ecrire (''Veuillez deux nombres entiers A et B '') ;
Lire (A, B) ;
R ← A mod B ;
Tant que R<>0 Faire
Début
A ← B ;
B ← R ;
R ← A mod B ;
Fin;
Ecrire (''Le PGDC de'', A, ''et'', B, ''='', A);
Fin
```

- **Exemple 4 :** Interprétez l'algorithme suivant :

```
Algo EQN_2nd_Degré ;
Var a, b, c, Delta, S1, S2, S : réels ;
Début
Ecrire (''Saisissez les valeurs de a, b et c dans l'équation ax² + bx + c  = 0'') ;
Lire (a, b, c) ;
Tant que a=0 Faire
Début
Ecrire (''Veuillez saisir une valeur de a différente de 0 !'') ;
Fin ;
Delta ← b² - 4*a*c ;
Si
Début si
 Delta>0 Alors
          S1 := ((-b – sqrt(Delta)/(2*a));
          S2 := ((-b + sqrt(Delta)/(2*a));
Ecrire (''Les solutions de l'équation sont :'', S1, ''et'', S2) ;
Fin si ;
Sinon
Début sinon
Si Delta=0 Alors
            S := ((-b/(2*a)) ;
 Ecrire (''La solution double est :'', S) ;
Sinon écrire (''Pas de solutions !'') ;
Fin sinon ;
Fin
```

Cet algorithme permet de résoudre une équation du second degré de type : $ax^2 + bx + c$.

- **Exemple 5 :** Même question de que la précédente :

```
Algo MultiAdd ;
Var i, a, b, R : entiers ;
Début
R ← 0 ;
Ecrire (''Saisissez les valeurs de deux nombres entiers a et b'') ;
Lire (a, b) ;
Si a=0 ou b=0 Alors écrire (''Le résultat de la multiplication est nul'') ;
Sinon
Début sinon
Pour i allant de 1 à b Faire
                        R ← R+a
Ecrire (''Le résultat de la multiplication est égale à :'', R) ;
Fin sinon ;
Fin
```

Cet algorithme, constitué de tests et de boucles (une boucle de type pour), nous permet de calculer la multiplication de deux nombres entiers en n'utilisant que des additions.

- **Exemple 6 :** Ecrire un algorithme qui calcule la somme des nombres entiers inferieurs à 100.

```
Algo somme_entiers_inf_cent ;
Var i, S : entiers ;
Début
i ← 1 ;
S ← 0 ;
Répéter
S ← S+i ;
I← i+1 ;
Jusqu'à  i=99 ;
Ecrire (''La somme des entiers inférieurs à 100 est :'', S) ;
Fin
```

- **Exemple 7 :** Ecrire un algorithme qui écrit le mot ''S0S''. (Le 0 dans S0S est un zéro)

```
Algo S0S ;
Var A, B : caractères
    C : str ;
Début
A ← ''S" ;
B ← ''0'' ;
C ← A&B&A ;
Ecrire (''C'') ;
Fin
```

- **Exemple 8 :** Détectez l'erreur dans l'algorithme suivant puis corrigez-la en réécrivant l'algorithme, tout en expliquant que fait cet algorithme.

```
Algo Bienvenue_2emeAnnée ;
Var A, B, C, D, Phrase : str ;
Début
A ← ''Bienvenue'' ;
B ← '' en'' ;
C ← '' 2eme'' ;
D ← '' année'' ;
Phrase ← A+B+C+D ;
Ecrire (Phrase) ;
Fin
```

L'erreur dans cet algorithme est flagrante, il est impossible de faire des opérations numériques sur des variables alphanumériques ! On utilise a la place des opérateurs alphanumériques, le plus utilisé est ''&'', et donc pour corriger cet algorithme il suffit de remplacer les signes ''+'' par ''&'', je vous laisse le loisir de réécrire cet algorithme correctement par vous-mêmes (oui je suis un flemmard).

Enfin cet algorithme, une fois écrit correctement bien sûr, va afficher le message suivant : Bienvenue en 2ème année.

- **Exemple 9 :** Pour ce dernier exemple, on va essayer de construire un algorithme qui va donner votre avis sur ce cours, nous tacherons d'utiliser des variables booléennes et ainsi nous aurons fait le tour de tout ce qui concerne les algorithmes (du moins à notre niveau actuel).

```
Algo Avis ;
Var A, B, C : str
    VRAI, FAUX : Bool ;
Début
A ← ''Oui'' ;
B ← ''Non'' ;
C ← ''Ce n'est pas vrai u_u'' ;
Ecrire (''Ce cours vous a plus Oui ou Non ?'') ;
Si
Début si
Lire (A) ;
Alors VRAI
Ecrire (''Merci !'') ;
Fin si ;
Si
Début si
Lire (B) ;
Alors FAUX
Ecrire (C) ;
Fin si ;
Répéter
Ecrire (C) ;
Jusqu'à
Lire (A) ;
Ecrire (''Merci !'') ;
Fin
```

Cet algorithme va vous demander d'entrer Oui ou Non selon que le cours vous a plus ou pas, si vous entrez Oui, l'ordinateur va afficher le message suivant : Merci !

Si vous entrez non, l'algorithme va afficher le message : Ce n'est pas vrai u_u

Tant vous entrez Non, l'algorithme continuera à afficher le message précèdent jusqu'à ce que vous entriez Oui ☺.

Pour finir je tiens à vous faire part de certaines remarques concernant les algorithmes que nous avons traités :

- Il est conseiller de laisser un point-virgule '';'' après chaque instruction (inutile de savoir pourquoi mais on ne sait jamais si le prof pourrait vous sanctionner à cause de ça).
- Le signe <> veut dire : est diffèrent de
- Le symbole sqrt (Square Root) signifie racine carré, l'ordinateur utilise ces symboles pour effectuer des opérations à l'aide de fonctions (comme la fonction racine carré).
- Le symbole := est utilisé pour affecter des résultats au cours des calculs

- Pour me PGDC, on a fait une suite de divisions jusqu'à ce que R=0 et le résultat c'est la dernière valeur de A qui n'a rien avoir avec ça valeur initiale (il ne faut pas confondre car on a fait des affectations en cours de route).
- Les algorithmes avec des variables booléennes sont rarement demandés mais nous avons au moins donné un exemple, on ne sait jamais, et pour ce type de variables on utilise des opérateurs logiques tel que OUI, NON, ET… Etc. (ne vous cassez pas trop la tête avec ça)
- Il faut impérativement respecter l'ordre des différentes instructions de l'algorithme, assurez-vous de ne rien avoir oublié et construisez votre algorithme de manière méthodique et logique comme nous l'avons appris ensemble ☺

Voici maintenant une lise de conseils qu'on a jugé utile de vous transmettre :

- En informatique, les définitions sont simples et faciles à retenir, donc essayez de les lire brièvement et de retenir les définitions susceptibles de pouvoir être posées en question à l'examen.
- Pour ce qui est du système binaire des pixels et des images, faut apprendre à faire les calculs c'est tout.
- Pour les algorithmes il faut savoir comment faire un algorithme demandé ou en interpréter.
- Essayez de bien comprendre toutes les parties du cours, peut être une seule lecture suffira et tachez de retenir les définitions, grosso modo c'est tout ce qu'il y a en informatique.

Voilà je pense que c'est tout ce qu'il y-a à savoir en ce qui concerne les algorithmes et l'informatique en général, j'espère que notre cours vous a plus, il y a certains qui vont peut-être le trouver un peu long mais nous nous somme assuré de fournir un cours complet et nous avons essayé de résumer au maximum le cours d'informatique tout en prenant soin d'y mentionner tout ce qu'il y a à savoir dans ce module, j'espère qu'il constituera pour vous un support pédagogique efficace qui vous aidera à décrocher la note maximale haut la main !

'' Nous souhaitons que vous aillez autant de plaisir à lire ce cours que nous a le faire '' l'équipe du module

Sources :

- Diaporamas des profs disponibles sur notre drive de la section D : goo.gl/oaEwOH
- Nous avons essayé d'exploiter nos informations personnelles, et le contenu des cours et des diaporamas des profs (ce qui justifie les similitudes que vous pourrez trouver sur notre cours et celui des profs), après tout ce sont eux qui vont faire le sujet de l'examen.
- Nous avons aussi jeté un coup d'œil sur le site : la-faculte.weebly.com
- Pour les algorithmes, nous avons tiré les exemples de séries de TD d'autres facultés (notamment la faculté de médecine d'Annaba).
- Nous avons essayé d'exploiter les documents d'autres facultés afin d'optimiser au maximum notre cours.

L'équipe :

- Hamid Rouane :
- **Facebook :** https://www.facebook.com/hamid.rouane
- **Email :** rouanehamid@gmail.com

- Youcef Guenane :
- **Facebook :** https://www.facebook.com/youyou.santos?fref=ts

Vous pouvez nous contacter avec nos coordonnées, personnellement vous pouvez toujours me trouver en ligne sur Facebook ou me contacter par email ou par tel, j'essayerais d'être aussi disponible que possible pour vous venir en aide, vous conseiller, et répondrai à vos questions à n'importe quel moment . Bon courage !!!

MERCI.